Jonathan Roderick
Onder Oz
Tyler Rather
EDITED BY: Aaron Curry

Experiment #1
Introduction to SPICE

Introduction:

This experiment is designed to familiarize the student with SPICE. SPICE simulations will be needed for prelabs and projects contained in this lab manual. Spice is an acronym for Simulation Program with Integrated Circuit Emphasis. SPICE is a computer aided design (CAD) tool that should be used to support design and should never be used in place of traditional design methods. Design by SPICE is a trap that many circuit designers fall into and it causes a designer to lose the insight that makes one truly successful. There are many different variations of SPICE that are produced by a few companies. Two of such versions are LTSPICE (provided for free by Linear Technology) and PSPICE (provided for free by Cadence). In the past, EE348 has taught students how to run HSPICE via the linux servers provided on campus. However, the graphical user interface (GUI) that the linux server has is much worse than that of freely distributed software. Therefore, this Lab shows how to use LTSPICE and PSPICE.

For the most part, SPICE code is universally accepted by SPICE engines. However, there are some differences in available commands and syntax from version to version. Therefore, if any problems are encountered by the version of SPICE you are using, you need to look at the reference manual to make sure you are using it properly. Links to the SPICE manuals for LTSPICE and PSPICE are provided in the "*references*" section at the end of this lab.

LTSPICE and PSPICE provide the user with the ability to generate SPICE netlists via schematics. While this tool is very useful, this lab will still focus on how to write a good SPICE netlist as opposed to create a schematic. This is because the ability to read and write a SPICE netlist is very valuable for electrical engineers.

One drawback of PSPICE is that its free demo version only allows the user to have up to 10 of the same type of transistor. However, since the labs in OHE230 only have PSPICE, students will have to use it during their labs.

First time LTSPICE setup on your computer:

1) Got to Linear Technology's website to download the .exe:
   http://www.linear.com/designtools/software/
2) Scroll down to where it says LTspice IV. To the right, click on the link that says, "Download LTspice IV." You do not need to register for a MyLinear account, just click on, "No thanks, just download the software."
3) After the download is complete, run the .exe file that you downloaded (LTspiceIV.exe). Accept the conditions and click "Install Now."
4) You are now ready to run LTspice

First time PSPICE setup on your computer:

1) Got to Cadence's website to download the .exe:
   http://www.cadence.com/products/orcad/pages/downloads.aspx#installer

2) Click on "PSPICE Schematics Installer".
3) You will need to fill out a form. The only field that needs to be valid is your email. After you complete the form, an email will be sent to you with a link to download the software.
4) Click on the link in the email and download and install the OrCAD Demo Software: "OrCAD 16.5 Demo Software (Capture and PSpice only)".
5) You are now ready to run PSPICE A/D lite.

NOTE: If you use a MAC computer, you can try running LTspice IV or PSPICE A/D lite using WINE. You will have to do your own research to figure this out. The LTspice manual talks about this procedure.

To simulate a circuit:
With SPICE you will need to first create a netlist. A netlist is a text file that describes the circuit and analysis desired. You then simulate the netlist, and after simulation you are able to view the results from the analysis requested.

For LTspice:
1) Go to File→Open
2) On the bottom of the pop-up where it says "Files of type:", change this to: Netlists (*.cir;*.net;*.sp).
3) Type in the desired file name, ending with .cir.
4) When LTspice asks you if you want to create a new file, click yes.

For Pspice:
1) Go to File→New→Text File

NOTE: When saving a file in PSPICE, save it as *filename*.cir. In order to simulate the PSPICE file, you must first close the file, and then reopen it.

Now you have a blank netlist file for you to write your netlist to be simulated. Note that every spice netlist starts with the name of the file, and ends with the .end statement.

Netlist:
This is a model of a typical SPICE input file.

Title (This must be the first line)
* Description of the circuit's function
Circuit Elements (three kinds)
  -Sources
  -Passive Circuit Elements
  -Active Circuit Elements
Model Statements
Analysis Requested
Output form Requested
.END (Before saving the file, make sure to press return once, and only once, after you type this statement)

The " * " denotes a comment, description, or statement added by the writer of the spice code. Any designated line with " * " at the beginning will be skipped by the complier and will not have an effect on the actual code. Any descriptions or comments can be omitted and the program will execute the exactly the same way. Any comments, descriptions, or statements are for nothing more than the convenience of the reader.

The "*Circuit Elements*" category describes all the physical components that are contained in the circuit schematic. "*Analysis Requested*" and "*Output form Requested*" are tools that are implemented in order to view a desired response from the circuit. The "*.end*" statement tells the compiler that the input file is finished. Don't worry too much about capitalizing letters, SPICE is case insensitive.

Circuit Elements Statements:

Each element is described to spice in the input file by an element statement. These statements contain the element name, the nodes where the element is located, and the physical characteristics of the element. The first letter in the element statement identifies what type of element the statement is for. The next two (two terminal device), three (three terminal device), four (four terminal device), etc., characters of the statement are for the node numbers that the terminals are connected to. The last part of the statement contains the physical model of the element. If certain device physical characteristics are left blank, then Spice has a set of default values that it will automatically use. Even though SPICE has some default values for a limited number of device characteristics, some parameters cannot be left blank. Table 1.1 contains a list of some basic elements and the letter that is used to identify them:

Table 1.1 Device Letter abbreviations.

First Letter	Element
C	Capacitor
D	Diode
E	Voltage-controlled voltage source (VCVS)
F	Current-controlled current source (CCCS)
G	Voltage-controlled current source (VCCS)
H	Current-controlled voltage source (VCCS)
I	Independent current source
L	Inductor
M	MOSFET
Q	BJT
R	Resistor
V	Independent voltage source

These elements fall into three different categories. Signal Sources, Passive Elements, and Active Elements.

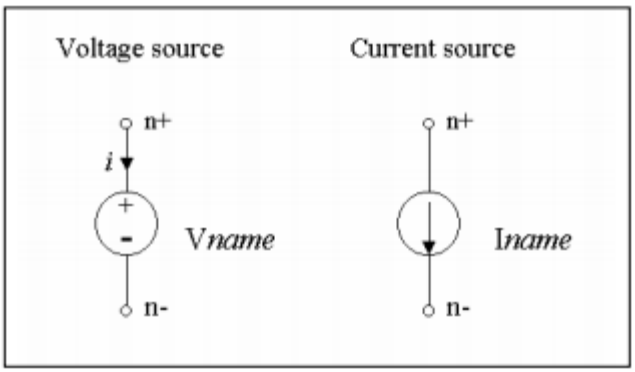

Figure 1.1. Independent Sources.

Independent Sources:

Three independent sources can be used in SPICE simulations. A DC source, a frequency-sweeping AC source, and time varying signals all can be modeled with standard SPICE signal sources. Each source can either supply current or voltage. Different types of sources are used for different types of analysis. Table 1.2 lists the different element statements for independent sources and the analysis type that is typically used with each.

Table 1.2 Source element statements

Type of Signal	Type of analysis	Element statement
DC Voltage	All types	V*name* n+ n- DC *value*
DC Current	All types	I*name* n+ n- DC *value*
AC Voltage	Frequency sweep	V*name* n+ n- AC *mag phase*
AC Current	Frequency sweep	I*name* n+ n- AC *mag phase*
SIN wave Voltage	Transient	V*name* n+ n- SIN (V_o V_a *freq* t_d *damping*)
SIN wave Current	Transient	I*name* n+ n- SIN (V_o V_a *freq* t_d *damping*)
Pulse wave Voltage	Transient	V*name* n+ n- PULSE (V_1 V_2 t_d t_r t_f *PW T*)
Pulse wave Current	Transient	I*name* n+ n- PULSE (V_1 V_2 t_d t_r t_f *PW T*)
Piecewise linear Voltage	Transient	V*name* n+ n- PWL(t_1,v_1 t_2,v_2...t_n,v_n)
Piecewise linear Current	Transient	I*name* n+ n- PWL(t_1,v_1 t_2,v_2...t_n,v_n)

Each element statement has an array of characters that represent different characteristics of the source. The characters are separated by a space so that the compiler knows that the user is done describing one aspect and starting another. In the element statements, the first letter depicts if the source delivers voltage or current with a V or I, respectively. The first letter is followed by the name of that source. Each source has to have a different name and they can be up to seven characters long. Next, the n+ and n- denote the node number of the positive and negative terminal, respectively. You will assign numbers (alphabetic characters are also valid in SPICE) to all the nodes in your circuit when writing an element statement. You may number (or name) them anyway you wish, with the exception of ground. Spice interprets node zero as the ground. The type of signal is identified after the node numbers. An example of each will be demonstrated.

DC:
Generic statement:
V*name* n+ n- DC *value*

Example:
V1 1 0 DC 10V

This is a DC voltage source with a value of 10V named "1" and its connected with its positive terminal at node 1 and its negative terminal at node 0 (ground). The name is arbitrary; you can call it anything you want.

AC:
Generic statement:
V*name* n+ n- AC (*mag phase*)

Example:
Vnew 5 0 AC (5V 2)

This statement describes a frequency-swept AC source called "new" that is connected to nodes 5 and 0, has a magnitude (mag) of 5V, and contains phase shift of 2 degrees. The phase can be left blank and SPICE will assume a zero degree phase shift.

SIN:
Generic statement:
V*name* n+ n- SIN (*Vo Va freq td theta phi ncycles(LTSPICE only)*)

Example:
Vinput 2 1 SIN (0V 5V 10e3 5e-3 0 0 0)

Vo is the initial voltage.
Va is the voltage amplitude.
freq is the frequency in Hz.
td is the time delay in seconds.
theta is the damping factor. This is used to apply an exponential decay to the sinusoid; theta is the decay constant in 1/seconds.
phi is the phase advance in degrees. Set this to 90 if you need a cosine wave form.
ncycles is the number of cycles of the pulse that should happen. Leave it as zero if you want ongoing pulses. THIS IS FOR LTSPICE ONLY.

This element statement describes a sinusoidal signal named "input" at it is connected between nodes 2 and 1. The signal has an initial voltage of 0V and a magnitude of 5V. It has a frequency of 10kHz and a time delay of 5milliseconds. The signal has no damping factor. If the statement does not contain values for td, damping, phase advance, or ncycles, then spice assumes a value of zero for all.

PULSE:
Generic statement:
V*name* n+ n- PULSE (*V1 V2 td tr tf PW T ncycles(LTSPICE only)*)
NOTE: tr and tf are swapped when declaring a pulse in PSPICE.

Example:
V50 15 0 PULSE (2 5 0 2e-3 4e-5 5 10 0)

V1 is the lower voltage value.
V2 is the upper voltage value.
td is the time delay in seconds.
tr is the time in seconds it takes for the pulse to rise.
tf is the time in seconds it takes for the pulse to fall.
PW is the pulse width of the peak value (or the time the pulse remains at the upper voltage value in seconds).
T is the time of one pulse period in seconds.
ncycles is the number of cycles of the pulse that should happen. Leave it as zero if you want ongoing pulses. THIS IS FOR LTSPICE ONLY.

The example details a pulse named "50". It is connected to nodes 15 and 0. The pulse has a lower voltage value of 2V and an upper value of 5V. The time delay is zero seconds, while it takes 2ms and 40µs for the pulse to rise and fall, respectively. The pulse spends 5 seconds at 5V and has a total period of 10 seconds.

EXP:
Generic:
V*name* n+ n- EXP(*v1 v2 td1 tau1 td2 tau2*)

Example:
V11 11 0 EXP(1 2 1u .5u 3u .1u)

V1 is the starting voltage.
V2 is the maximum voltage.
td1 is the time in seconds to wait at the starting voltage before changing to the maximum voltage.
tau1 is the time constant for the change from the starting voltage to the maximum voltage.
td2 is the time in seconds to wait (from time t=0) before changing back to the starting voltage.
tau2 is the time constant for the change back to the starting voltage.

The example details an exponential voltage named "11". It is connected to nodes 11 and 0. It starts at 1 volt and waits 1µs before it starts to rise to 2V with a time constant of .5µ. At 3µs it stops rising and starts to fall back to 1V with a time constant of .1µ.

PWL:
Generic:
V*name* n+ n- PWL (*t1,v1 t2,v2…tn,vn*)

Example:
V100 25 10 PWL(0,0 3e-3,5V 6e-3,5V 10e-3,-3V)

To use a piecewise linear source, state a voltage and the corresponding time you wish the source to reach that value. In the example above at time 0, the source will have a value of 0V. The source will then perform a linear increase for the next 3ms till it reaches 5V. The next point given has the same voltage value, so the source will have the value of 5V for the next 3ms. The source will then follow a linear degradation till it reaches –3V four milliseconds later.

Graphs of the time varying signals:

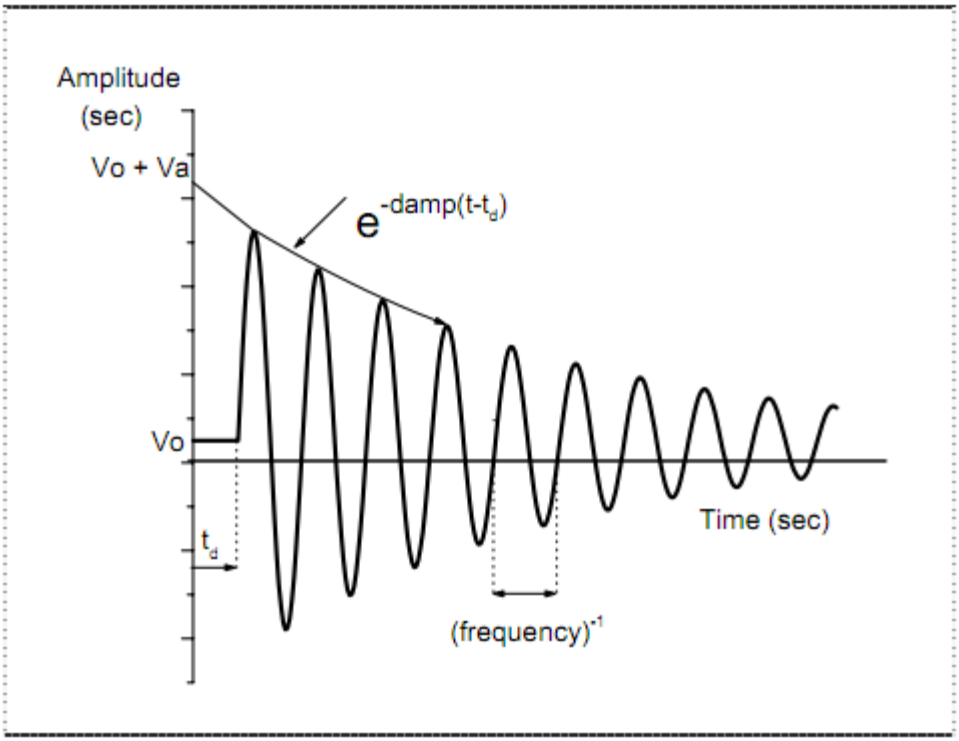

Figure 1.2. A SIN wave created in SPICE.

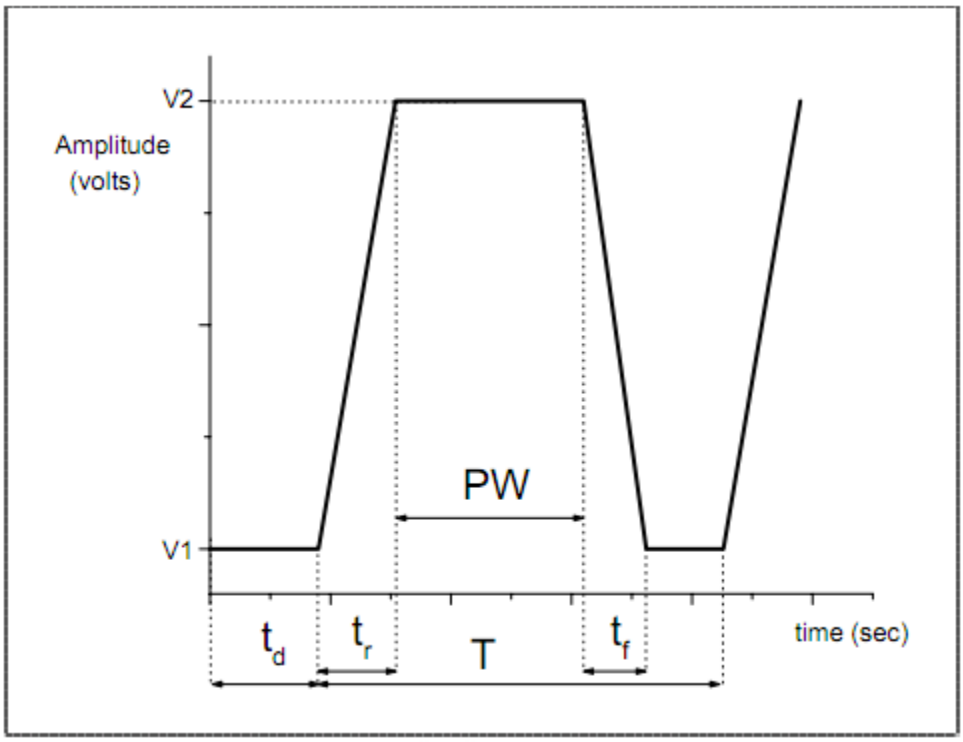

Figure 1.3. A Pulse wave created in SPICE.

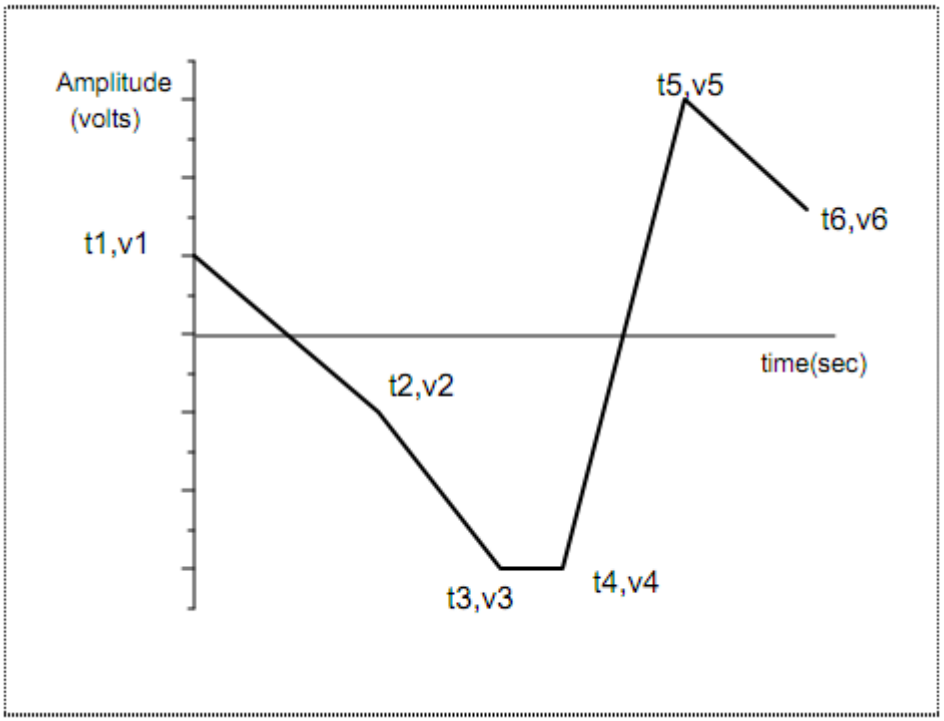

Figure 1.4. A piecewise linear signal created in SPICE.

Dependant Sources:
Generic statement:
E*name* n+ n- p+ p- *A*

Example:
Eone 2 3 1 0 50

There are four nodes in the dependant source command. The first two nodes of the voltage controlled voltage source (VCVS), n+ and n-, represent the node numbers for the positive and negative ports of the output. The second set of node numbers represents the positive and negative nodes of source's reference voltage. The gain of the VCVS is indicated by the value of A. In the example, the controlling, or reference, voltage of the VCVS is between nodes 1 and 0, while it has a gain of 50. The output of the VCVS is connected between nodes 2 and 3. The value of the output is dependent on the controlling voltage by a factor of the gain.

When dealing with units, there are certain scale-factor abbreviations that HSPICE will recognize. The acceptable abbreviations for HSPICE are listed in table 1.3.

Table 1.3 Scaling abbreviations

Spice Abbreviation	Metric Prefix	Multiplication factor
T	tera	10^{+12}
G	giga	10^{+9}
Meg	mega	10^{+6}
K	kilo	10^{+3}
M	milli	10^{-3}
U	micro	10^{-6}
N	nano	10^{-9}
P	pico	10^{-12}
F	femto	10^{-15}

SPICE does not require you to label the units, however here are the acceptable unit abbreviations.

Table 1.4 Unit abbreviations.

Spice Abbreviation	Units
A	Amps
Degree	Degree
F	Farad
H	Henery
Hz	Hertz
Ohm	Ohms (Ω)
V	Volts

Caution:

If not caught, SPICE has a really fatal flaw. If you notice, the abbreviation for Farads and femto are both " F ". This can be a cause for real heart burn. If you were to label your capacitance value 1F thinking this represents one farad, you would soon find out that spice interprets this as 1 femto farad (1e-15 Farads). To be safe, just leave the units off when dealing with capacitance.
Another common mistake is mixing up mega with milli. For example if you want to make a resistance of 1 mega-ohms and use m instead of meg, you will create a resistance of 1 milli-ohms.

Passive elements:

Passive element statements are very similar to DC sources. The first letter indicates the passive element that is being used. The letters for a standard resistor, capacitor and inductor are listed in table 1.1. A schematic diagram of each is shown below. The schematic diagrams accompanied by a generic statement, an example statement and a brief explanation.

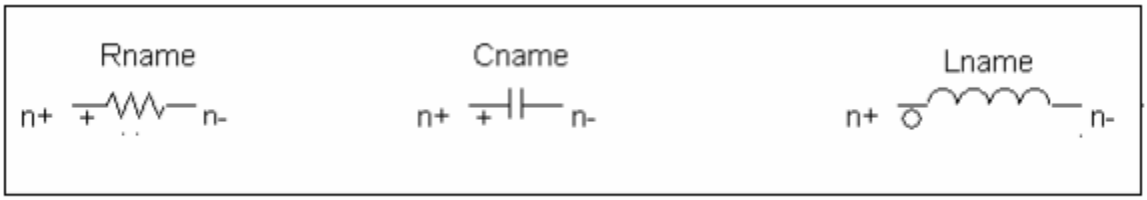

Figure 1.5. Schematic symbols of a resistor, capacitor, and inductor.

Resistor:

Generic:
R*name* n+ n- *value*

Example:
R1 5 0 10k

The example depicts a resistor named "1", which is connected between nodes 5 and 0 (ground).
It has a value of 10k Ohms.

Capacitor:

Generic:
C*name* n+ n- *value* [IC=*initial condition*]

Example:
Cload 50 0 10u [IC=0.5V]

This example is a capacitor named "load" connected at nodes 50 and 0 (ground) and it is 10 microfarads. This capacitor also has a 0.5V initial condition. This means that the capacitor has an initial voltage at time equal to zero. If the "[IC=*initial condition*]" part is left off, SPICE will assume that the initial voltage on the capacitor is zero volts. **Caution: Remember the unit problems with capacitors stated earlier.**

Inductor:
Generic:
L*name* n+ n- *value* [IC=*initial condition*]

Example:
Lfeedback 50 0 1

The inductor is modeled similar to the capacitor, except the initial condition is in terms of current as opposed to voltage. This particular inductor has a 0A initial condition.

Active elements:
The three basic active elements needed for this class are the diode, BJT and the MOSFET. The element statements for active devices are really similar to passive. The fundamental difference is that the BJT and MOSFET are three and four terminal devices, respectively. The physical characteristics can also be a little more complicated. Other than that, active elements follow the same basic element statement format. The first letter identifies the device type, the next three statements are for the node numbers (only two in the case of the diode), and the last part of the statement contains the physical characteristics. Since active devices have complicated physical characteristics a ".Model" Statement is used. An example of a diode, BJT and MOSFET are all show below. To better illustrate this technique, they will have an accompanying explanation.

Diode:

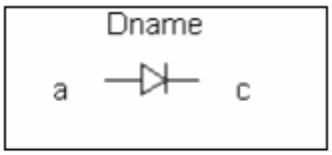

Figure 1.6. A schematic symbol of a diode.
a represents the anode terminal.
c represents the cathode terminal.

Generic:
*(under device section)
D*name* a c *model_name*
*(Under Model Statement section)
.Model *model_name* D [*certain parameter#1=value certain parameter#2=value* … etc]

10Example:
*(under device section)
Drec 2 3 fermi
*(under Model Statement section)
.Model fermi D [Is=150pA n=1.2]

This example is of a diode named "rec". It is connected between nodes number 2 and 3. This diode references its physical characteristics from a .Model statement called "fermi". In this particular example

the only parameters dictated to Spice are Is and n. More parameters can be specified if desired. Table 1.5 is a list of typical parameters specified for a diode and the default setting for each.

Table 1.5 Diode model parameters.

Spice Abbreviation	Diode Characteristics	Units	Default
Is	Saturation current	Amps	1×10^{-14}
Rs	Ohmic Resistance	Ohms	0
n	Ideality factor	Unit less	1
BV	Reverse-bias breakdown voltage	Volts	∞
IBV	Reverse-bias breakdown current	Amps	1×10^{-10}

BJT:

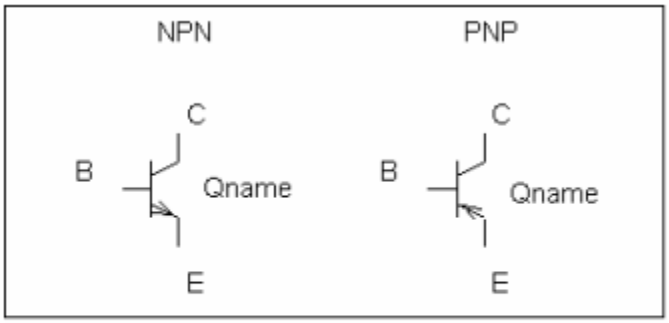

Figure 1.7. Schematic symbol for BJTs.
B is the Base terminal
C is the Collector terminal
E is the emitter terminal.

The BJT element statement is very similar to the diode. The main difference is that it has three terminals instead of two. Other than that, the structure of the statement is very similar. An example of a NPN is done below.

Generic:
*(under device section)
Q*name* C B E *model_name*
*(under Model Statement section)
.Model *model_name* NPN [*certain parameter#1=value, certain parameter#2=value* … etc]

Example:
*(under device section)
Q1 3 14 5 normal
*(under Model Statement section)
.Model normal NPN [Is=3e-14, Bf=150, Vaf=30V]

The example statement describes a BJT with its collector connected at node 3, base connected at node 14, and emitter connected at node 5. It references the ".model" statement called "normal" for all the physical characteristics. "normal" indicates to Spice that the BJT called "1" is an NPN transistor and gives some associated characteristics. The ".Model" statement is easily modified to change the transistor to a PNP type. Simply replace the NPN with PNP in the statement.
Table 1.6 give some basic characteristics variables for a typical BJT and the default values associated with each.

Table 1.6 BJT model parameters.

Spice Abbreviation	Model Parameter	Units	Default
Is	Saturation current	Amps	$1\ 3\ 10^{-16}$
Bf	Forward Gain	Unit less	100
Rb	Base resistance	Ohms (Ω)	0
Rc	Collector resistance	Ohms (Ω)	0
Re	Emitter resistance	Ohms (Ω)	0
VAf	Forward Early voltage	Volts	∞

MOSFET:

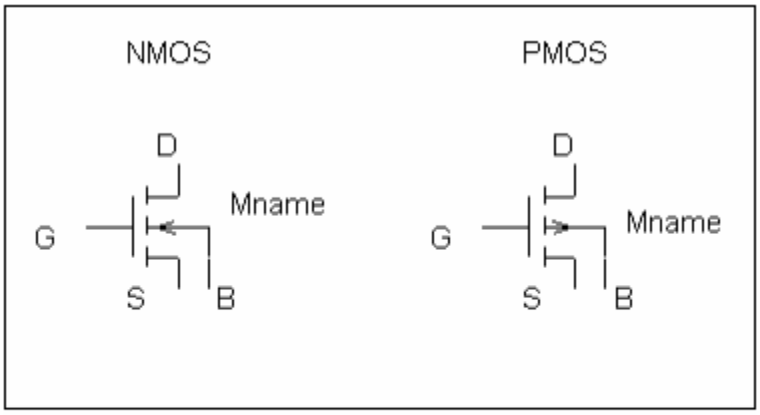

Figure 1.8. Schematic symbol of MOSFETS
G is the Gate terminal
D is the Drain terminal
S is the Source terminal
B is the Bulk terminal

The element statement structure for the MOSFET is basically the same as the BJT, but with one more terminal. An example is given and explained below.

Generic:
*(under device section)
M*name* D G S B *Model_name* L=*value* W=*value*
*(under Model Statement section)
.Model *model_name* NMOS [*certain parameter#1=value, certain parameter#2=value* … etc]

Example:
*(under device section)
M55 3 15 0 0 typical L=1u W=20u
*(under Model Statement section)
. Model typical NMOS [kp=10u Vto=1.5 lambda=0]

The MOSFET described by the example is called "55". Its drain is connected to node 3, while the gate, source, and bulk are connected to nodes 15, 0(ground), 0(ground) respectively. SPICE sources the .Model called "typical" for its physical characteristics. The new addition of L and W describe the physical dimensions of the device. L stands for the length of the channel, while W stands for the width. The type of MOSFET is identified in the ".model" statement. In this example the MOSFET is a NMOS, but replacing the "NMOS" part in the statement with "PMOS" will change the type of MOSFET from NMOS to PMOS. A table of other typical MOSFET characteristics and their default values are listed in table 1.7.

Table 1.7 MOSFET model parameters.

Spice Abbreviation	Model Parameter	Units	Default
Level	MOSFET Model number		Level 1
kp	Transconductance coefficient (μCox)	A/V^2	20μ
Rs	Source resistance	Ohms (Ω)	0
Rd	Drain resistance	Ohms (Ω)	0
Vto	Threshold Voltage	V	0
lambda	Channel Length Modulation	V^{-1}	0
gamma	Body effect	$V^{1/2}$	0
phi	Surface potential ($2\phi_f$)	V	0.6

Command Statements:
Once you have entered all the element statements in the input file, you have to enter the command statements. These command statements tell SPICE exactly how you want to simulate the circuit and present the data. As opposed to element statements, all command statements begin with a period.

Analysis type:
Once the element statements are complete, the analysis that is to be done needs to be specified to SPICE. This is done with an analysis request statement. When creating an input file, you must decide from the start what type of analysis you want to perform. As listed in table 1.2, some independent sources only work with certain types of analysis. There are four basic types of analysis to choose from: DC operating point, DC sweep, AC frequency response, and transient response. An example of each analysis request statement is done below.
LTspice only lets the user have one analysis type work for each simulation. Therefore, if you want to have multiple analysis types for a given circuit, you will need to simulate it multiple times. If you want to view the operating point for an analysis, this is saved in a file named *filename*.op.raw.

DC operating point requested:
.OP

DC sweep:
Generic statement:
.DC *source_name starting_value ending_value step_value*

Example:
.DC power 0V 5V 0.1V

A DC sweep is done for many reasons. For example, you would use a DC sweep if you wanted to see the different responses of a circuit for different biasing conditions. In the example the ".DC" analysis command instructs the source named "power" in the circuit to start at 0V and increase its values by increments of 0.1V until it reaches a value of 5V, then it stops. You would then use a specific output request to see the response (this will be discussed later).

AC frequency response:
Generic:
.AC *type number_of_points frequency_start frequency_stop*

There are three different types of AC frequency responses. The difference deals with how the points are taken. You can specify the following types: DEC (points are spaced logarithmically by decade), OCT (points are spaced logarithmically by octave), or LIN (points are linear spaced).

Example of the three different types:
.AC DEC 100 2k 1e6
.AC OCT 1000 2k 1e6
.AC LIN 50 500 550

The ".AC" command does not need the name of the source. An AC source is the only source that allows for this type of analysis, so HSPICE will implement this analysis to the AC source in the circuit automatically. The first ".AC" command depicts a statement that performs a frequency sweep that takes 100 points per decade. The analysis starts at 2kHz and ends at 1MHz. The other two statements follow the same format. Following the ".AC" command are statements that indicates the way points are taken and the number of points taken, respectively. Finally, the starting and ending frequencies are listed.

Transient response:
Generic:
.TRAN *step_time stop_time*

Example:
.TRAN 10n 10m

This example simulates the circuit for 10ms and takes data points every 10ns. The ".TRAN" analysis is preformed when the response of the circuit with time is desired. For example, a transient response request would be used if you wanted to see the time it takes a capacitor to charge or discharge. **Caution: For correct data collection, make sure that the smallest step time in the transient command is equal to**

or smaller then the fastest change in your signal. You must also make sure that the transient time complements the source, or you will not be able to see the simulated results.

Transfer Function:
Generic:
.TF V(*node*) *source*

Example:
.TF V(out) Vin
.TF V(2,1) Vin
.TF I(Rout) Vin

These examples all compute the DC small-signal transfer functions for different outputs with respect to Vin. The first example finds the transfer function from the node called "out" to Vin. The second example finds the transfer function from the differential voltage from nodes 2 to 1 to Vin. The third example finds the transfer function from the current in the resistor called "out" to Vin.
NOTE: in PSPICE, when the output variable is a current, it is restricted to be the current through a voltage source.

Output requested:
With PSPICE you can request the data collected to be in the form of data points or as a plot. This is done with a ".PRINT" or ".PLOT" statement. The type of print or plot statement depends on what type of analysis you requested. Table 1.8 lists these statements.

Table 1.8 Output data type requests.

Output requested	Analysis	Spice Command
Print data points	DC	.PRINT DC *output*
	AC	.PRINT AC *output*
	TRAN	.PRINT TRAN *output*
Plot data points	DC	.PLOT DC *output* [(*lower_ limit upper_ limit*)]
	AC	.PLOT AC *output* [(*lower_ limit upper_ limit*)]
	TRAN	.PLOT TRAN *output* [(*lower_ limit upper_ limit*)]

You will be able to view the data collected in the output file that is created once you simulate the input file through PSPICE. The *output* gives information on what you are measuring (you can measure voltage, voltage difference, or current through any element) and the node number where you want to observe the activity of the circuit. You many request data from multiple outputs, just separate the "output" requests with a space. LTSPICE does not create an output file. If you want to view the operating point, again it can be found in *filename*.op.raw.
Table 1.9 lists different "*output*" statements that may be used to measure voltage, voltage difference, or current in a circuit.

Table 1.9 Data collection methods for .print and .plot commands.

Measuring	***"output form"***
Voltage at a specific node	V(*node_number*)
Voltage difference between two nodes	V (*node_number1*, *node_number2*)
Current through any element	I(*Element_name*)

Some examples are done below.

Probe: (For Pspice only)
Generic:
.Probe *output*

Example:
.Probe V(1) I(Rout)
.Probe

The first example will write the results of the voltage at node 1 and the current through the resistor named "out". The second example will write all the node voltages and element currents in the circuit. These results can be viewed with the wave viewer provided by PSPICE.

Print: (For Pspice only)
Generic:
.PRINT *type output*

Example:
.PRINT DC V(out)

This example will print the values of the node voltage called "out" from the DC sweep in a table in the output file.

Plot: (For Pspice only)
Generic:
.Plot *type output*

Example:
.PLOT AC I(R10)

This example will plot the values of the current through the resistor called "10" from the ac sweep in the output file.

NOTE: Plot and Print are methods for viewing results from a SPICE analysis in the output file. This is the standard method for an HSPICE netlist. However, PSPICE and LTSPICE have a wave viewer that allows the user to plot the results from analysis. These are much better for viewing purposes. Thus,

.probe can be used with PSPICE (no command is needed for LTSPICE) and the built in wave viewers can be used.

After you have entered all the elements, analysis, and output request you must end the input file with an ".END" statement. Make sure to press return after you have typed your .END statement. Not doing this sometimes hang ups SPICE.

Graphical Interface Plots:

In order for LTSPICE to plot a desired trace, first simulate your circuit by going to: Simulation→Run, or by clicking the run button ( ). After the simulation is done, the waveform viewer will automatically open. In order to add a trace go to:
Plot Settings→Add Trace. A window that looks similar to the following will come up:

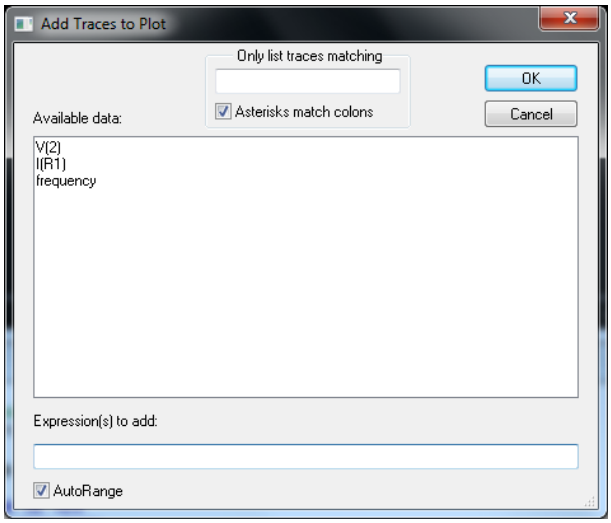

Figure 1.9. Window that appears in LTSPICE when adding traces.

From this window you can add traces by selecting them and clicking "OK".

In PSPICE, if you have the .PROBE command entered, the waveform viewer will automatically open after the simulation is done. First, simulate your circuit by going to Simulation→Run, or by clicking on the run button ( ). Then, you can add traces by going to: Trace→Add Trace. A window that looks similar to the following will come up:

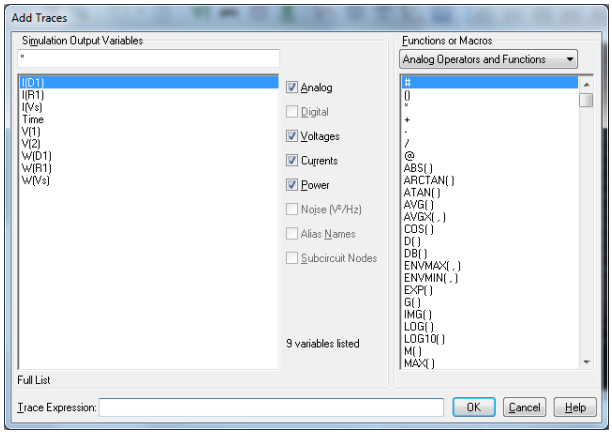

Figure 1.10. Window that appears in PSPICE when adding traces.

From this window you can add traces by selecting them and clicking "OK".

Trace Expressions:

When adding traces in LTSPICE or PSPICE, you can add expressions of traces. This is very helpful when you want to see mathematical properties of node voltages and element currents. Expressions can be written in the "Trace Expression" line in the above two windows when traces are added.

The following table lists the functions available for real data in LTSPICE.

Table 1.10 List of allowed functions for real data in LTSPICE.

Function Name	Description
abs(x)	Absolute value of x
acos(x)	Arc cosine of x
arccos(x)	Synonym for acos()
acosh(x)	Arc hyperbolic cosine
asin(x)	Arc sine
arcsin(x)	Synonym for sin()
asinh(x)	Arc hyperbolic sine
atan(x)	Arc tangent of x
arctan(x)	Synonym for atan()
atan2(y, x)	Four quadrant arc tangent of y/x
atanh(x)	Arc hyperbolic tangent
buf(x)	1 if x > .5, else 0
ceil(x)	Integer equal or greater than x
cos(x)	Cosine of x
cosh(x)	Hyperbolic cosine of x
d()	Finite difference-based derivative
exp(x)	e to the x
floor(x)	Integer equal to or less than x
hypot(x,y)	sqrt(x**2 + y**2)
if(x,y,z)	If x > .5, then y else z
int(x)	Convert x to integer
inv(x)	0. if x > .5, else 1.
limit(x,y,z)	Intermediate value of x, y, and z
ln(x)	Natural logarithm of x
log(x)	Alternate syntax for ln()
log10(x)	Base 10 logarithm

max(x,y)	The greater of x or y
min(x,y)	The smaller of x or y
pow(x,y)	x**y
pwr(x,y)	abs(x)**y
pwrs(x,y)	sgn(x)*abs(x)**y
rand(x)	Random number between 0 and 1 depending on the integer value of x.
random(x)	Similar to rand(), but smoothly transitions between values.
round(x)	Nearest integer to x
sgn(x)	Sign of x
sin(x)	Sine of x
sinh(x)	Hyperbolic sine of x
sqrt(x)	Square root of x
table(x,a,b,c,d,...)	Interpolate a value for x based on a look up table given as a set of pairs of points.
tan(x)	Tangent of x.
tanh(x)	Hyperbolic tangent of x
u(x)	Unit step, i.e., 1 if x > 0., else 0.
uramp(x)	x if x > 0., else 0.
white(x)	Random number between -.5 and .5 smoothly transitions between values even more smoothly than random().

For complex data, the functions atan2(,), sgn(), u(), buf(), inv() uramp(), int(), floor(), ceil(), rand(), min(,), limit(,), if(,,), and table(...) are not available. The functions Re(x) and Im(x) are available for complex data and return a complex number with the real part equal to the real or imaginary part of the argument respectively and the imaginary part equal to zero. The functions Ph(x) and Mag(x) are also available for complex data and return a complex number with the real part equal to the phase angle or magnitude of the argument respectively and the imaginary part equal to zero. The function conj(x) is also available for complex data and returns the complex conjugate of x.

For adding expressions of traces in PSPICE, all the available functions are listed on the right-hand side of the add trace window. Simply click on the trace on the left, then click on the expression you want to be performed on the trace on the right.

Example of an input file

```
Example input file
*
Vs 2  0 SIN ( 0V 1V 10k)
R1 2 1 10k
D1 1 0 diode
*
.MODEL diode D (Is=2E-13 N=1.1)
.OP
.TRAN .01m 1m
.PLOT  TRAN V(1)
.END
```

Note: SPICE is case insensitive.

A Schematic of the example input file.

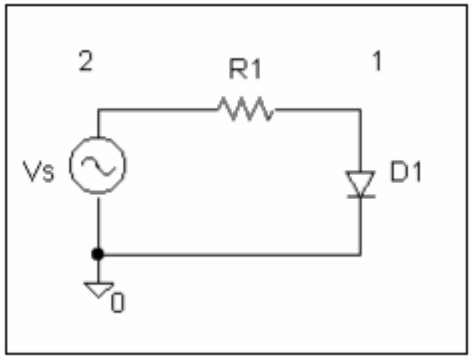

Figure 1.11 Schematic of example input file.

The netlist and plots of nodes 1 and 2 as seen in LTSPICE

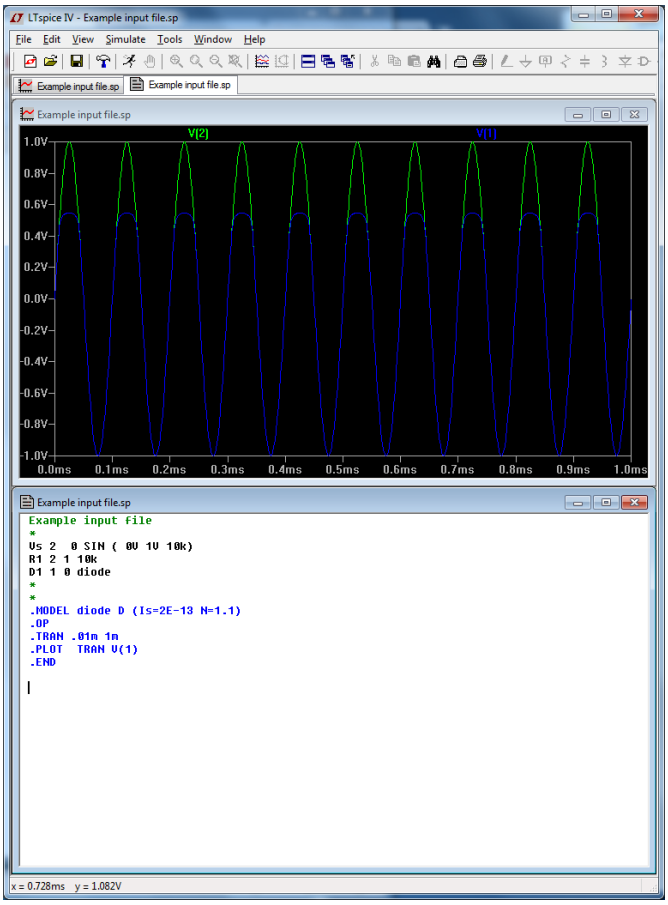

Figure 1.12. A screen shot of the netlist from the example input file and plots of nodes 1 and 2 taken using LTSPICE

This lab is no way intended to be an LTSPICE or PSPICE manual. This introduction only covers the basics that a student will need to run LTSPICE and PSPICE. Other published material, such as the ones in the reference reading, on SPICE should be consulted for a better understanding.

Reference Reading

1. Gordon W. Roberts and Adel S. Sedra. SPICE. Second edition. New York: Oxford Press, 1997.

2. Marc E. Herniter. Schematic Capture with MicroSim Pspice. Third edition. Upper Saddle River, New Jersey: Prentice Hall, 1998.

3.LTSPICE IV user manual. http://ltspice.linear.com/software/scad3.pdf

4. PSPICE user manual. http://www.electronics-lab.com/downloads/schematic/013/tutorial/PSPCREF.pdf

Lab exercises

(1) The topology below is the circuit from the first homework, problem #1. Verify your results from problem #1, part h) of the homework using SPICE. Please turn in all graphs that prove the results you obtained. Be sure to label the graphs to indicate what Av(0), Rout, τ_{RC}, and $\omega_{3\text{-dB}}$ are. Last of all, turn in your SPICE netlist.

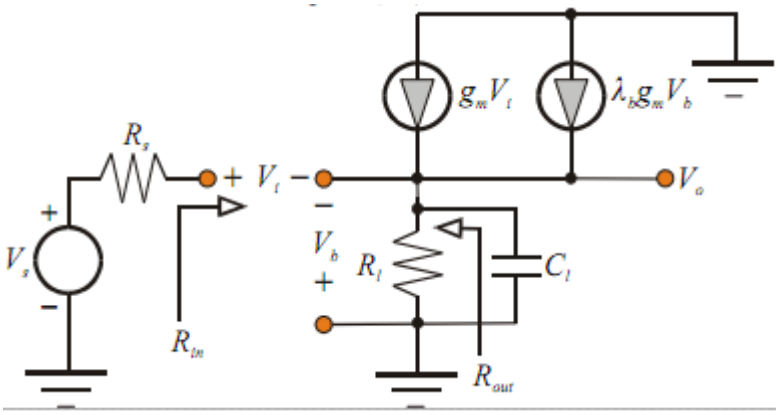

Figure 1.13. The circuit used for homework 1, problem #1

(2) The topology below is the circuit from the first homework, problem #2. Please verify the exact results you found in problem #2, part h using SPICE. Please turn in all graphs that prove the results you obtained. Also note that you must label graphs to indicate what Rin, Rout, 3dB bandwidth, or Av(0) are. Last of all, please turn in your SPICE netlist.

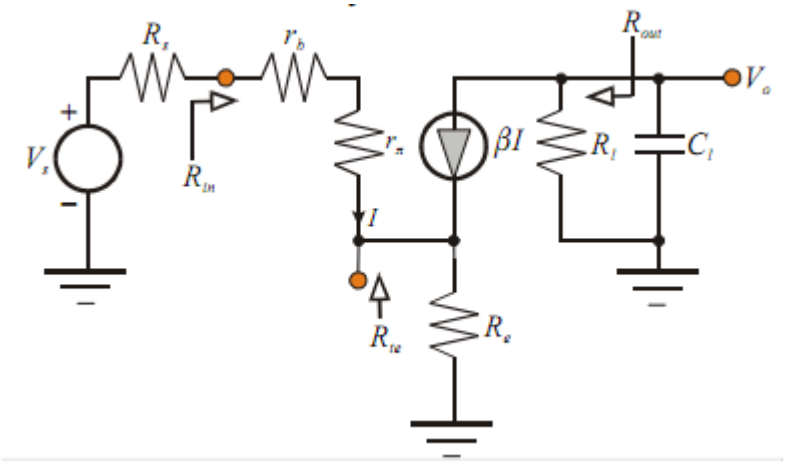

Figure 1.14. The circuit used for homework 1, problem #2

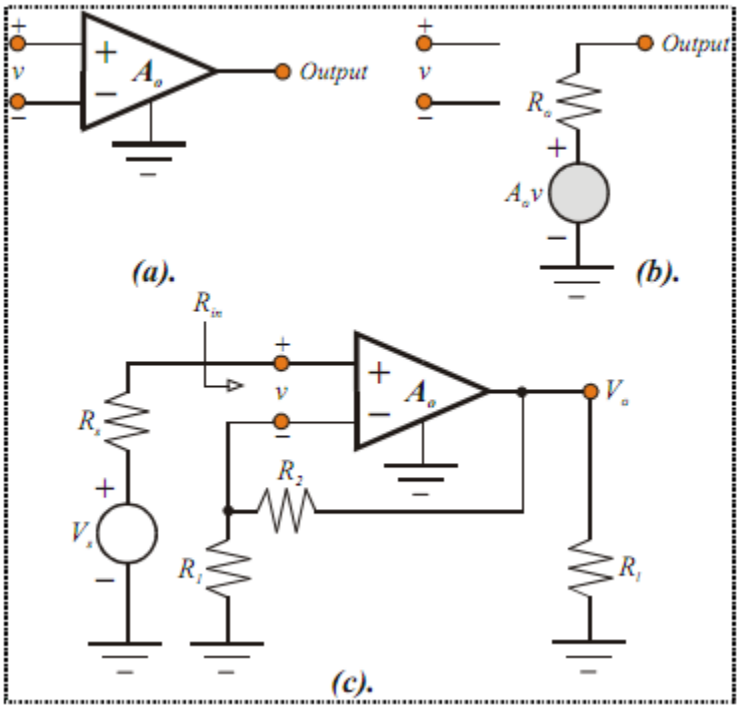

Figure 1.15. The circuit used for homework 1, problem #5

(3) The circuit in figure 1.13 is the problem# 05.

(a) Given Rs 50 ohms, R2 800 ohms R1 80 ohms Rl 50 ohms Ro 20 ohms Ao 20 Simulate figure1.13 c to show what Vo/Vs is.

(b) Given Rs 50 ohms, R2 800 ohms R1 80 ohms Rl 50 ohms Ro 2 ohms Ao 2000 Simulate figure1.13 c to show what Vo/Vs is.

(c) Please explain in a sentence or two what caused the difference in between part a and b

General Report Format Guidelines

1. Introduction

   Explain what the lab is about. Describe the circuits being built in terms of structure and purpose. Also talk about what is being investigated.

2. Procedure

   Step by step talk about what was done and show diagrams of the circuits.

3. Data

   Present all data taken during the lab. It should be organized and easy to read.

4. Discussion

   Discuss the results you obtained. What significance is there in the results? How do they help your investigation? Explain the meaning; the numbers alone aren't good enough.

5. Questions

   Answer all the questions in the lab.

6. Conclusion

   Wrap up the report by giving some comments on the lab. Do the results clearly agree with what the lab was trying to teach? Did you have any problems? Suggestions?